CodeWarrior Plug-in for Freescale HCS12(X)

Device Initialization User Manual

Help version 2.9

Copyright 2010 Freescale Semiconductor, Inc.
PROCESSOR EXPERT is trademark of Freescale Semiconductor, Inc.

CONTENTS

1. Introduction 4

1.1. Features and Benefits 4
1.2. Basic Terms and Definitions 4
1.3. Quick Start 5
1.4. Rapid Application Development Tools 6

2. User Interface 8

2.1. Main Menu 8
2.2. Device Initialization Window 9
2.3. Inspector Dialog Box 12
2.3.1. Inspector Items 14
2.4. Error Window 15
2.5. Code Generation Options Dialog Box 16

3. Using the Tool 18

3.1. Peripheral Initialization Components 18
3.2. Code Generation And Usage 19
3.3. Saving and Restoring the Design 21
3.4. Defining Interrupt Service Routines 21
3.5. Changing The CPU 22
3.6. Converting Code Warrior Project To Use The Device Initialization 23
3.7. Converting Device Initialization Project To Processor Expert 24

4. Tutorials 26

5. Help Revisions History 27

1. Introduction

Device Initialization is a fast, easy, and user-friendly way to configure and generate a CPU peripheral initialization code.

1.1. Features and Benefits

The key features of the Device Initialization tool are:

- Graphical user interface with CPU package, peripherals and pins.
- User-friendly access to the initialization setup of the CPU peripherals.
- Initialization code generator.
- User can select Assembly or C format for the generated code.
- Built-in detailed design specifications of the Freescale CPUs.
- Initialization options are automatically transformed into the peripheral control registers values.
- Easy to view control register values resulting from the parameter settings. Changes are immediately highlighted.
- Changes in the peripheral control registers values are transformed back into the component parameters.

The key benefits of the Device Initialization tool are:

- Easy to learn design environment.
- User friendly names of the peripheral features - no need to explore manuals for the control register details.
- Easy changes in initialization.
- Possibility to reuse the individual peripheral setup for other designs.
- No need to generate code to see the resulting peripheral control registers values.

1.2. Basic Terms and Definitions

Component - Peripheral Initialization component - is a component that encapsulates initialization of a peripheral. A component can be configured using the Inspector. See *3.1 Peripheral Initialization Components* for details.

CPU component - Component configuring the CPU core parameters. It cannot be removed from the project.

Design - All components (the CPU component and the Peripheral Initialization components) with their customized parameters.

Inspector - Window that allow to view and configure parameters of the selected component.

Internal peripherals - Internal peripherals of the CPU (ports, timers, A/D converters, etc. usually controlled by the CPU core using special registers).

ISR - Interrupt Service Routine - code which is called when an interrupt occurs.

Module - Source code module. Could be generated by the tool or created by the user and included in the project (user module).

Peripheral settings - Initialization parameters for an individual peripheral. These settings can be saved and restored from the file.

Popup menu - This menu is displayed when the right mouse button is pressed on some graphical object.

Properties - Parameters of the component. Property settings define the internal peripheral initialization state that will be converted to generated initialization code during the code generation process.

Target CPU - CPU derivative used in a given design.

1.3. Quick Start

This section describes how to create a simple design, configure a device, generate initialization code and use it in your application.

Step 1. Create an Empty Design

Use the Project Wizard to create a new project. It can be invoked by clicking on the **Create New Project** button in the Startup screen or using the menu command **File > New Project...**.

Follow the step by step settings and in the **Rapid Application Development Options** page and select the **Device Initialization** option.

Step 2. Configure Peripherals

The Device Initialization window shows the CPU package with the available internal peripherals.

Click on a peripheral field to configure its initialization. A new peripheral initialization component is automatically created and the Inspector dialog window is displayed. It allows the user to view and change the parameters.

Use the Inspector dialog window that appears to setup initialization parameters of the peripheral as per your requirement and confirm it by clicking on the OK button. Use the same steps for all peripherals you wish to setup.

Step 3. Generate Code

Click the Generate Code button in the Device Initialization window.

Within the Options dialog box, that appears immediately, **select the desired initialization code format: C or Assembly.**

Please note that on ColdFire CPUs, the assembly code generation is not available.

Within the Options dialog box, that appears immediately, you can specify the name of the output files you wish to generate (you can also keep a default name 'MCUInit') using the field *Generated module name*.

For more details on generated code, see the *3.2 Code Generation And Usage*.

Step 4. Use the Generated Code

The MCU_init initialization function call is automatically placed at the beginning of the main routine. Start writing your application code after this initialization function call.

1.4. Rapid Application Development Tools

Two tools are available in the CodeWarrior IDE for rapid application development: Processor Expert and Device Initialization. Both tools have many advanced features that lead to development cycle acceleration.

Features Comparison

Feature	Processor Expert	Device Initialization
Easy to use graphical IDE	✔	✔
Interactive design specifications of Freescale MCUs	✔	✔
Generated code	Peripheral Drivers	Initialization code only
Generated code languages	C	C or Assembly
Peripheral Init Components	✔	✔
Low Level Components	✔	✘
High Level Components	✔	✘
Project Configurations	✔	✘
User-friendly linker parameter file configuration	✔	✘
Generated code changes tracking	✔	✘
Timing settings in time units (such as seconds and bauds)	✔	✘
Free components library on the web	✔	✘
User components creation	✔	✘

Device Initialization

Device Initialization provides a fast and easy way to configure and generate an initialization source code for the CPU. It contains only one set of components: Peripheral Initialization Components. The code initializing the peripheral can be generated for use in Assembler or C.

Processor Expert

Processor Expert generates drivers in the C language that allows a hardware-independent access to CPU peripherals. It contains large library of components called Embedded Components. Embedded Components encapsulate the initialization and functionality of embedded systems basic elements, such as CPU core, CPU on-chip peripherals, standalone peripherals, virtual devices and pure software algorithms. These facilities are interfaced to the user using properties, methods, and events, such as objects in Object Oriented Programming (OOP).

Note: Processor Expert Embedded Components were formerly called Embedded Beans.

Description of the Embedded Components on different levels of abstraction:

- **High Level Components** - Basic set of components designed carefully to provide functionality to most of the microcontrollers in the market. An application built from these components can be easily ported to another microcontroller supported by the Processor Expert. This group of components can provide comfortable settings of a desired functionality such as time in ms or frequency in Hz, without the user knowing about the details of the hardware registers.
- **Low Level Components** - Components dependent on the peripheral structure that allow the user to benefit from the non-standard features of a peripheral. The level of portability is decreased due to a different component interface and the component is usually implemented only for a CPU family offering the appropriate peripheral. However, there is still possible to easily configure device's features and use a set of methods and events.
- **Peripheral Initialization Components** - Components on the lowest level of abstraction. An interface of such components is based on the set of the peripheral control registers. These components cover all features of the peripherals and are designed for initialization of these peripherals.

For more details, please see the Processor Expert documentation.

2. User Interface

Menu

Device Initialization menu is integrated into the CodeWarrior IDE menu. See *2.1 Main Menu* for details.

Windows and Dialog Boxes

The Device Initialization user interface consists of the following windows that are integrated in the CodeWarrior IDE:

- **Device Initialization** - Main window graphically showing CPU package, structure and components connected to the internal peripherals. It allows the user to easily to add components related to a specific peripheral to the design. See *2.2 Device Initialization Window* for details.
- **Inspector** - Window that allows the user to setup peripheral initialization components parameters. See *2.3 Inspector Dialog Box* for details.
- **Error window** - Window with errors, warning messages and hints from design checking and code generation. See *2.4 Error Window* for details.
- **Code Generation Options** - Dialog box with design and code generation related settings. See *2.5 Code Generation Options Dialog Box* for details.

2.1. Main Menu

Device Initialization menu is integrated in the CodeWarrior IDE main menu.

The menu contains the following commands:

- **Initialize Device** - Creates a new design for the currently opened CodeWarrior project or opens an existing one, if it already exists on the disk. If the design is already opened, only the Device Initialization window is activated. See *2.2 Device Initialization Window* for details.
- **Backup Device Settings** - Invokes a file selection dialog box and if the user confirms the selection, it saves all design settings using the selected file name. See *3.3 Saving and Restoring the Design* for details.
- **Restore Device Settings** - Invokes a file selection dialog box and if the user confirms the selection, it loads all design settings from the selected file name. See *3.3 Saving and Restoring the Design* for details.
- **Update PE from Package** - Installs a patch or update from the .PEUpd file.
- **Options** - Invokes the Options dialog box that contains the design-related settings with default values for next code generations. See *2.5 Code Generation Options Dialog Box* for details.
- **Generate Code *{designname}.mcp*** - Initiates the initialization code generation process. See *3.2 Code Generation And Usage* for details.
- **View Report**
  - **Project Settings** - Generates a new document containing XML data with parameters and values of all components in the design.
  - **Register Settings** - Generates a new document containing XML data with all registers (and their values) that will be set within generated code.
  - **Interrupt Usage** - Generates a new document containing XML data with all interrupts allocation by ISR routines.
  - **Pin Usage** - Generates a new document containing XML data of all pins allocated by the device initialization design.

Help Menu

The help pages links is integrated in the Help menu of the CodeWarrior under the **Help > Device Initialization** submenu.

- **Device Initialization help** - displays the main help file.
- **Device Initialization Tutorial** - opens a tutorial showing the device initialization basics.
- **List of all supported CPUs** - displays the page with the list of CPUs supported in the current installation.
- **List of all supported Peripheral Initialization Components** - displays the page with the list of CPUs supported in the current installation.
- **View Read Me and Revision History** - shows the document containing details related to the current version and the history of revisions and changes of the Device Initialization tool.

2.2. Device Initialization Window

The Device Initialization window is the main window of the Device Initialization design. The window contain the **design control buttons** at the top of the window and **working area** that allows the user to browse and configure the CPU peripherals. To open the Device Initialization window, select **Device Initialization > Initialize Device** in the menu bar.

Design Control Buttons

The window contains the following control buttons:

- **Select CPU package** - Lists available packages from the current target CPU. From the list of packages, the user can choose the one to be used in the design. See *3.5 Changing The CPU* for details.
- **Generate Code** - Invokes the Code Generation dialog allowing the user to generate the initialization code according to the current settings. See *3.2 Code Generation And Usage* for details.
- **Backup** - Invokes a file selection dialog and if the user confirms the selection, saves all design settings using the selected file name. See *3.3 Saving and Restoring the Design* for details.
- **Restore** - Allows to restore the whole design from a file. File selection dialog is invoked immediately. See *3.3 Saving and Restoring the Design* for details.
- **Help** - Opens the help page.

Work Area

This area allows the user to configure the CPU peripherals by adding the Peripheral Initialization Components. See *3.1 Peripheral Initialization Components* for details.
The following view modes are available:

- **Package View of the CPU mode**: Configure a peripheral by clicking on the peripheral field.
- **Alphabetically ordered peripherals list mode**: Configure a peripheral by clicking on its name or icon.

Control Icons

- - Rotates the CPU by 90 degrees.
- - Zoom-in - Increases the package diagram size and detail level.

- - Zoom-out - Decreases the package diagram size and detail level.
- - Activates the Package View mode.
- - Activates the Peripherals List mode.

Package View Mode

This is the default view mode for the design. In this mode, the window contains:

- Peripherals available on the CPU and their allocation by components.
- Pins and their allocation by components.
- Useful information that is available in the status line if the mouse cursor is placed on pin, component or peripheral.

The following information about each pin is displayed on the package:

- Pin name (either default or user-defined).
- Icon of a component that uses (allocates) the pin.
- Direction of the pin (input, output, or input/output) symbolized by blue arrows, if a component is connected.

Some pins cannot be used by the user because they are allocated by special signals such as power signals, external or data buses. Special pins are indicated by special blue icons, for example . The allocation of pins by special signals could be affected by the CPU component properties.

Peripheral List

This mode shows the alphabetically ordered list of all CPU peripherals and their allocation. Unallocated peripherals have a gray icon.

Hints

A hint appears when the mouse cursor is placed on a specific item:

A pin hint contains:

- pin number
- pin name
- owner of the pin (component that allocates it)
- short pin description

Note: The pin hint is available only in the package view mode

Component icon hint contains:

- component type
- component description

Component Pop-up Menu

This menu appears when the user right-clicks on a component icon button. It contains the following commands:

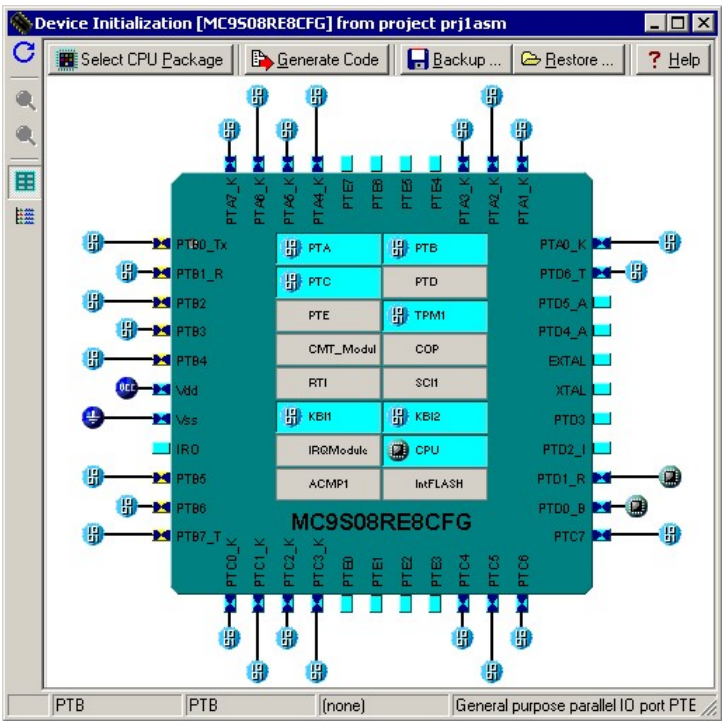
- **Disable Peripheral Initialization** - Removes the selected component with all its settings from the design.
- **Help on Component** - Opens the help pages related to the selected component.

General Pop-up Menu

This menu appears when the user right-clicks anywhere in the Device Initialization window. It contains the following command:

- **Help on Device Initialization Window** - Shows the help page.

Sample Screenshot



2.3. Inspector Dialog Box

The Inspector dialog box allows the user to modify parameters of components configuring internal peripherals of the target CPU. (For more details on the peripheral components, please see *3.1 Peripheral Initialization Components*.)

Inspector dialog box consists of two panels:

- **Component Parameters** - Contains the parameters that influence the initialization of the selected device.
- **Register Details** - Contains the values of the individual control registers that is set by a generated initialization code.

Control Buttons

The buttons description from left to right:

-

- Removes the currently opened peripheral initialization component from the design. This command is not available for the CPU component.
- - Opens the file selection dialog box and saves parameters to the selected file.
- - Opens the file selection dialog box and restores parameters from the selected file.
- - Opens a help page with the description of the component parameters.
-

- Enables/disables the Register Map column in the Register details panel (see below).
-

- Saves the initialization. parameters and closes the window
-

- Cancels changes and closes the window.
- - Opens the help page of the Inspector window.

Component Parameters Panel

Component Parameters panel contains the four columns:

- **Item status**
  - green checkmark - Item setting is correct.
  - red exclamation - Item setting is not correct. Items that cause errors or warnings are written in magenta color. See description in the last column or the Error Window.
  - plus or minus - Item is a group of settings that can be expanded or collapsed.
  - light background - Item is version specific and is displayed only for CPU derivatives that support it. These items cover the special capabilities of the CPU that are not present for all CPUs.
- **Item names** - Items that are to be set are listed in the second column of the inspector. Groups of items describing certain features can be collapsed or expanded by double clicking on the first line of the group.
- **Selected settings** - Settings of the items are made in the third column. See chapter *2.3.1 Inspector Items* for list of item types.
- **Setting status** - Current setting or an error status may be reflected on the same line, in the rightmost column of the inspector. The error is also displayed in the item's hint.

A parameter can be presented as read-only and the user cannot change its content. Such read-only values are shown in gray.

Register Details

This panel shows values of individual control or data registers related to the currently selected CPU peripheral and reflects the settings in the *Component Parameters* panel.
On some peripherals there may be present an *Additionally modified registers* section within this panel that lists the registers that are initialized but belong to a different peripheral.
The following two types of rows can be found in this panel:

- **Whole register content**

The row contains the following four columns:

- **Status field** - Contains plus or minus - each register can be expanded or collapsed by clicking the icon to show or hide individual bits.
- **Name** - Specifies name of the register according to the CPU datasheet.
- **Address** - Specifies address of a register.
- **Init. Value** - Specifies the initialization value of a register or bit computed according to the settings in the *Component Parameters* panel. **This is the last value written by the initialization function to the register.**
  Note: For some registers, the value read from the register afterwards can be different than the last written value. For example, some interrupt flags are cleared by writing 1. Please see the CPU manual for details on registers behavior.

  If the row is active, the value can be changed using the keyboard. The field also contains a radix button (H,D,B) that allows to switch between Hexadecimal, Decimal and Binary formats of the value.
  The value that contains undefined bits is always shown in binary format.

- **Register Map** - Graphical representation of the initialization value. The colors represent states of individual bits:
  (White) = value 0
  (Black) = value 1
  (Gray) = bit has no meaning for initialization of the peripheral (reserved, read-only or related to another peripheral) or the value of the bit is undefined.
  Presence of this column depends on the **View Register Map** checkbox.

- **Individual bit of the register**

The row contains the bit value icon (0 or 1) and the name of the bit. Bits are sorted from the highest to lowest weight. The bit rows of a register can be shown or hidden using the and icons in the status field of the register row.

Changes Highlighting

The user can immediately watch the impact of the changes in the *Component Parameters* panel on the CPU peripheral registers and bits values. The registers influenced by a last settings change are highlighted in green

(see the figure below for example).

Figure 2.4 - The register affected by the parameter change

The changes made to a register values are automatically analyzed and transformed into the component parameters values. Such parameters changes are highlighted the same way the register changes are.

The changes highlighting works only if the component has been set-up correctly before the change is made and the new setup is correct and no error reported in the component settings. If there is an error in the component settings, the after-reset values are shown.

2.3.1. Inspector Items

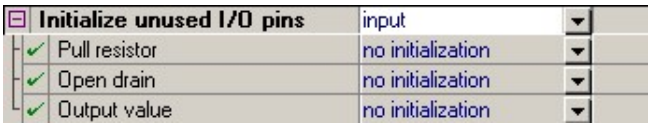
The following types of the items are found in the Inspector:

- **Boolean Group** - A group of settings controlled by this boolean property. If the group is enabled, all the items under the group are valid; if it is disabled, the list of items is not valid. Clicking the + sign shows/hides the items in the group but doesn't influence the value or validity of the items.

- **Boolean yes / no** - The user can switch between two states of the property using the round icon .

- **Enumeration** - Selection from a list of values. If the user clicks on the arrow icon ( ), a list of the possible values for the property is offered.

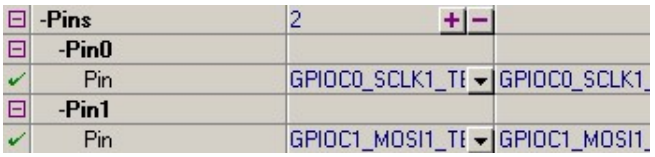
- **Enumeration Group** - A list of items. Number of visible (and valid) items in the group depends on the chosen value. Clicking the arrow icon ( ) shows a list of the possible values of the property. Clicking the + sign shows/hides the items in the group but doesn't influence the value or validity of the items.



- **Group** - A list of items that can be expanded/collapsed by clicking the plus/minus icon or by double clicking at the row. Values of the items in the group are untouched.

- **Integer Number** - The user can insert a number of the selected radix. Radix of the number can be switched using the icons D H B (D = Decimal ,H = Hexadecimal, B = Binary). Only reasonable radixes are offered for the property. If the radix switching icon is not present, Processor Expert expects the decimal radix.

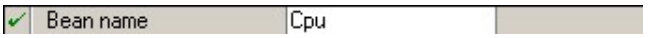
- **List of items** - A list of items may be expanded/collapsed by clicking on the plus/minus button at the left side of the row or by double clicking on the row. The user may add/remove an item by clicking on the plus/minus button. The items in the list can be arranged using the related pop-up menu commands.



- **Peripheral selection** - The user can select a peripheral from the list of the available peripherals. The peripherals that are already allocated have the component icon in the list. The properties that conflicts with the component settings have the red exclamation mark.

- **Real Number** - The user can insert any real (floating point) number.

- **String** - Allows to enter any text or value.



2.4. Error Window

This window is automatically displayed if there are any errors, warnings or hints found during:

- automatic design checking
- code generation

Some errors are found right after inconsistent or incorrect data is entered, others during the code generation of a design. The single messages mention the component where the error is found. If an error concerns two components, the error is attributed to both components.

To improve the readability of the Error window right-click the Error window to display a pop-up menu and delete the desired tools or code generation errors, warnings, and hints.

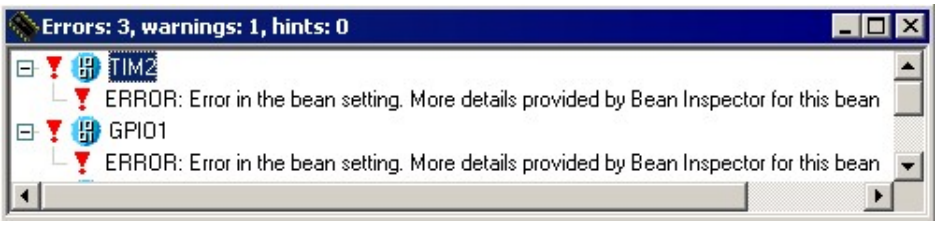

Errors: 3, warnings: 1, hints: 0
TIM2
ERROR: Error in the bean setting. More details provided by Bean Inspector for this bean
GPIO1
ERROR: Error in the bean setting. More details provided by Bean Inspector for this bean

Figure 2.15 - Processor Expert Error window

Pop-up Menu

The pop-up menu, invoked by right-clicking, contains the following items:

- **Delete All Tool Errors, Warnings and Hits** - Removes all tool errors, warnings and hints listed in the error window
- **Delete All code generation errors, warnings and Hints** - Removes all code generation errors, warnings and hints listed in the Error window
- **Copy to Clipboard** - Copies the whole content of the window as a text to the clipboard.
  Note: This command can be very useful in the case you need to contact our support personnel with a component setup issue.
- **Help** - Displays documentation

2.5. Code Generation Options Dialog Box

This dialog box is invoked at each code generation or using the **Options** command from the main menu The user can specify the options influencing the format and placement of the generated code. Options are divided into two groups using tabs.

Basic Options

Basic options include the following options and option groups:

- **Generated file types** available:
  (For more information on generated files, please see the chapter *3.2 Code Generation And Usage*)
  - **Relocatable Assembler** - Generates the relocatable code in the assembly language. This option is not available for the absolute assembly projects.
  - **Absolute Assembler** - Generates an absolute code in assembly language. This option is supported only if it was selected in the CodeWarrior Project wizard.
  - **C** - Generates the code in the C language. This option is not available for the assembly projects.
- **After Generation**
  - **Save and add files to the project** - The files produced by Processor Expert are named using the value of the *Generated Module Name* field. The files are automatically added to the *Generated Code* folder of the Code Warrior project.
  - **Create file and do not add to project** - The code will be generated into the newly created untitled editor files.
- **Generated module name** - Specifies name of the files where the initialization code will be generated.

Advanced Options

The following options modify the generation of code:

- **Generate register modification only if initialization value does not match reset state** - This option does not affect the registers writable only once (for example CONFIGx) nor the registers placed in FLASH (for example MORx).
- **Generate comments about register modification** - Source code will contain comments with descriptions of the registers values.
- **Generate interrupt vector table** - Interrupt vector table content will be generated to the output file.
- **Generate interrupt service routine templates** - Tool will generate an empty interrupt routines definitions for all enabled interrupts according to the components parameters. See *3.4 Defining Interrupt Service Routines* for details.
- **Generate initialization of registers writable only once** - These registers can be written only once after reset due to technological or safety reasons. This options enables the generation of initialization code for these registers.
- **Generate initialization of register placed in FLASH** - Initialization of these registers will be done during the programming of the application code to the FLASH memory.
- **After code generation show description how to use the generated files** - If this option is enabled, a dialog box with the short description of the generated modules and their usage is displayed after every code generation.

Common Options

- **Show this dialog every time before code generation** - Using this check-box, the user can enable or disable appearance of this dialog box before every code generation.

3. Using the Tool

The sub-chapters describe basic principles and tasks related to device initialization.

- Peripheral Initialization Components
- Code Generation And Usage
- Saving and Restoring the Design
- Defining Interrupt Service Routines
- Changing The CPU
- Converting Code Warrior Project To Use The Device Initialization
- Converting Device Initialization Project To Processor Expert

3.1. Peripheral Initialization Components

A Peripheral Initialization Component is an object that provides a human-readable interface to the initialization parameters of a selected on-chip CPU peripheral. Parameters of the Peripheral Initialization Component represent the settings of all peripheral registers in the clear tabular form. Names of the Peripheral Initialization Components are *Init_<peripheral>* and they are specific for each CPU family.

Adding Component

Components can be added to the design using the Device Initialization window. Click on the unallocated peripheral to add a new component. The new component is preset to work with the selected device. Inspector dialog box appears allowing the user to configure parameters of the component.

Component Parameters and Device Registers

Every component contains a group of parameters (properties) that describe the desired initialization state of the device. These parameters are automatically transformed to values of control registers related to the selected peripheral. Inspector shows both - component parameters and resulting registers values.

Component parameters are grouped to several groups by type. The following groups are commonly present in peripheral initialization components:

- **Settings** - Common peripheral parameters.
- **Pins** - Configuration of the pins related to the peripheral.
- **Interrupts** - Configuration of the interrupts and interrupt service routines related to the peripheral. See *3.4 Defining Interrupt Service Routines* for details.
- **Initialization** - Parameters related to the peripheral initialization.

CPU components

A CPU component is the component that configures the parameters of the CPU core, such as clock source and frequency and power-saving capabilities. The CPU component is always present in design and cannot be removed. CPU component contains the following parameter groups:

- **Clock Settings** - Configuration of the CPU timing
- **Internal Peripherals** - Configurations of the peripherals not supported by separate components and settings that can be shared among components.
- **CPU Interrupts** - Configuration of the interrupts related to the CPU core.

Modifying Components Settings

Parameters of existing components can be configured using the Inspector dialog which can be opened by clicking on the component's icon in the Device Initialization window.

3.2. Code Generation And Usage

Starting The Code Generation

To generate code:

1. Click the *Generate Code* button in the Device Initialization or select the **Generate Code {design name}** command from the main menu.
   A dialog box with the code generation options appears, if it is not disabled by the '*Show this dialog every time before code generation*' check-box. See *2.5 Code Generation Options Dialog Box* for details.

2. Click the **Generate** button in the Options dialog box. The source code initializing the CPU registers according to the specified component parameters is generated.

After a successful code generation, the dialog box appears with a basic information on generated code and its usage. This dialog box can be enabled or disabled in the Options dialog box. See *2.5 Code Generation Options Dialog Box* for details.

Generated Code

During the Code Generation process the Device Initialization tool generates the initialization code for the CPU peripherals allocated by the components. The generated initialization code is split into two functions - *MCU_init_reset* and *MCU_init*.

- **MCU_init_reset** - the function contains initialization of the memory configuration. By default, this function is called after reset.
- **MCU_init** - the function contains the reset of the initialization code. The user shall call this function at the start of application code to initialize peripherals.

The generated module consists of two files:

- **Implementation file** containing the code of the initialization function(s) and optionally the interrupt vectors table.
- **Interface file** containing the declarations that can be referenced by the user. This file is generated only if the files are stored to a disc (see below).

Depending on the *After generation* option, the files can be stored to a disk and added to the current project or just shown in the editor as untitled files. See *2.5 Code Generation Options Dialog Box* for details.

Device Initialization tool can generate the following types of initialization code:

- **Relocatable Assembler** - The implementation file has the extension **.asm** and the interface file has the extension **.inc**. This option is not available for the absolute assembly projects.
- **Absolute Assembler** - The implementation file has the extension **.inc** and must be included at the end of the user module, where address for code is selected (using ORG). Absolute assembler is supported only if it was selected in the CodeWarrior Project wizard.

- **C language** - The implementation file has the extension **.c** and the interface file has the extension **.h**.

A default name for the generated module is '*MCUInit*'. An Initialization code format, the generated module name, and other code generation options, can be configured in the Options dialog box. See *2.5 Code Generation Options Dialog Box* for details.

User Changes in the Generated Code

If the content of generated modules is written to the disk, it always overwrites the existing files during every code generation. As a result all the user modification are discarded with the following exceptions:

- user definitions (or generally any other source code) in .C (or .asm) placed in the special comment marks. In case of C language it looks like:
```
/* User declarations and definitions */
  User source code...
/* End of user declarations and definitions */
```
- content of interrupt service routines that are assigned to any interrupt(s) in the peripheral initialization components. See *3.4 Defining Interrupt Service Routines* for details.
- unused interrupt service routines (no component contains their name). They do not disappear but they are moved to the end of the file. The generated comments before and after the ISR must not be modified for this feature to work properly.

Note: No user changes in the .h file (or .inc in case of assembly language) are preserved. The file is always overwritten.

Using the Generated Code

To call MCU_init function from the main file, it's necessary to do the following modification in your code:

- Device initialization by default generates an interrupt vectors table containing all interrupt vectors (it can be disabled in the Options dialog box, see chapter *2.5 Code Generation Options Dialog Box* for details). Existing interrupt vector definitions have to be removed or commented out to avoid a conflict.

  For details on configuring interrupt vectors in Device Initialization please see the chapter *3.4 Defining Interrupt Service Routines*.

- Add a call to the generated *MCU_init* function at the beginning of the application main routine. The newly created projects already contain this line.
  Note: The prototype or external declaration of the MCU_init function or a command including the interface file with this declaration should be present in the module where the MCU_init is called. In a new project, it is already there in the main file.

- Use MCU_init_reset as the Reset Vector (newly created projects use it by default).

For step-by-step instructions on how to convert the existing C or assembly project to use the DeviceInitialization please see the chapter *3.6 Converting Code Warrior Project To Use The Device Initialization*.

3.3. Saving and Restoring the Design

Automatic Saving

All parameters and settings of the device initialization design are stored in the file with the extension **.iPE** and the **same name as the CodeWarrior project file**. It is also located within the same directory.

The design is saved after each successful code generation with the option **Save and add files to project**, so it reflects the state of the code in the generated modules.

The design is automatically loaded when the project created with the Device Initialization option is opened.

Numbered Archive Files

When the design is automatically saved, the previous content of the saved file is not overwritten. It is renamed to a new name with the number appended to the end and stored in the same directory. This number is automatically incremented after every save. Previous design versions data, stored in numbered files, can be manually restored using a manual restore function (see the following paragraph for details).

Manual Backup and Restore

The user can manually save or restore all settings to or from a .iPE file on the disk. This file can be used in another project, archived or for example sent by an e-mail.
To perform manual backup or restore, you can use the **Backup** and **Restore** buttons in the Device Initialization Window or use the main menu commands **Backup Device Settings** and **Restore Device Settings**. The restored settings override all current settings and they are lost. The user is warned about it and has to confirm the restoration process. See chapters *2.2 Device Initialization Window* and *2.1 Main Menu* for details.

Closing Device Initialization Window

If there are any changes in the design or the code is not generated yet, a dialog box asking for confirmation to save design appears after closing the Device Initialization window. If the user confirms this dialog box, the file name selection dialog box appears allowing the user to choose a name for the file.

3.4. Defining Interrupt Service Routines

Some Peripheral Initialization components allow the initialization of an interrupt service routine. Interrupt(s) can be enabled in initialization code using appropriate parameters that can be found within the group *Interrupts*.
After enabling, the specification of an Interrupt Service Routine (ISR) name using the *ISR name* property is required. This name is generated to Interrupt Vector table during the code generation process. See *3.2 Code Generation And Usage* for details.
Please note that if the ISR name is filled it is generated into the Interrupt Vector Table even if the interrupt property is disabled.

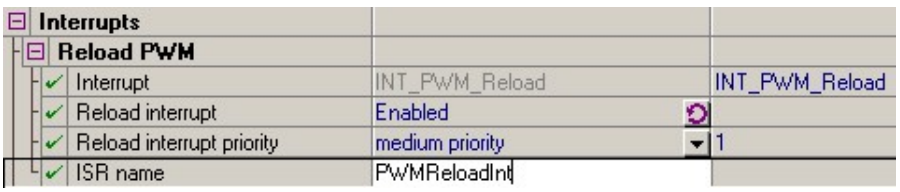


Figure 3.1 - Example Of The Interrupt Setup

Enabling or disabling peripheral interrupts during runtime has to be done by the user's code.

Interrupt Service Routines Code

The ISR with the specified name has to be declared according to the compiler conventions and fully implemented by the user. Declarations of the ISRs that do not exist yet can be generated automatically during the code generation process into the generated module if the option **Generate interrupt service routine templates** is enabled. See *2.5 Code Generation Options Dialog Box* for details.

The contents of interrupt service routines, written by the user, that are assigned to any interrupt within the components parameters is protected against being overwritten during the code Generation process. In case the interrupt service routine is not assigned to any interrupt, it's moved to the end of the file.

Warning: *The user is responsible for synchronizing ISR names in the code and ISR names specified in components. If an ISR is renamed, the name has to be changed in the component(s) where this ISR name is assigned and vice versa. This has to be done before next code generation. Otherwise the newly specified ISR would not be found and the existing ISR with an old name will be treated as unassigned, that is, it will be moved to the end of file.*

3.5. Changing The CPU

Changing CPU package

The type of the CPU package can be changed using the *Select CPU Package* button in the Device Initialization window. See *2.2 Device Initialization Window* for details.

Switching the Project to a Different CPU Derivative

To switch to a different CPU derivative for the project, select **Project Change MCU / Connection** from the menu bar in the CodeWarrior IDE.

Components Assignment

If some peripherals of the MCU set by components are not supported by the new MCU derivative the project is switched to, a dialog box with a list of the unsupported items is shown. The user is asked to confirm that these items will be removed from the design.

3.6. Converting Code Warrior Project To Use The Device Initialization

This chapter guides the user through a conversion from a plain C or assembly project to the project using the Device Initialization plugin and a peripheral initialization code generated by this tool.

The following steps should be done to convert the project:

1. Open the project you want to convert.
2. Select the **Device Initialization** > **Initialize Device** main menu command. Confirm the dialog window with the question '*Do you want to add a new iPE device setting?*' by clicking the 'Yes' button.
   A Device Initialization window with a CPU package appears.
3. Configure the peripherals and generate the initialization code by using the **Generate code** button. See chapters *3.1 Peripheral Initialization Components* and *3.2 Code Generation And Usage* for details.
4. Open and modify the main file of the project to reference and call the MCU_init subroutine the following way:
   (This subroutine will contain an initialization code generated according to the peripheral configuration after the 'Generate code' button is pressed. See 3.2 Code Generation And Usage for details.)

   - ***For Relocatable Assembly Project :***

     Find the part of the code including the derivative information and add the bold marked line:

     ```
     ; Include derivative-specific definitions
                 INCLUDE 'derivative.inc'
                 XREF MCU_init
     ```

     Then find the main routine start and add the MCU_init call at the place you want the peripherals to be initialized :

     ```
     ; code section
     MyCode:     SECTION
     main:
     _Startup:
                 ...
                 ; Call generated Device Initialization function
                 JSR    MCU_init
     ```

   - ***For Absolute Assembly Project :***

     Find the part of the code including the derivative information and add the bold marked line:

     ```
     ; Include derivative-specific definitions
                 INCLUDE 'derivative.inc'
                 INCLUDE 'MCUInit.inc'
     ```

     Then find the main routine start and add the MCU_init call at the place you want the peripherals to be initialized :

     ```
     _Startup:
                 ...
                 ; Call generated Device Initialization function
                 JSR    MCU_init
     ```

- ***For C Project :***

  Add the MCU_init function declaration and its call to the main C module by adding the bold marked lines into the code:

```
void MCU_init(void); /* Device init function declaration */
...
void main(void) {
  MCU_init(); /* call Device Initialization */
...
```

5. Remove, comment or modify the existing code that conflicts with the generated initialization code.
   *Note: The Device Initialization generates the complete interrupt vector table. Any interrupt declarations in the .prm file (lines starting with **vector** keyword) or elsewhere need to be removed or commented out and all interrupt need to be configured by using the Device initialization. Please see the chapter 3.4 Defining Interrupt Service Routines for details.*
6. Build the application. The peripheral initialization code can be anytime re-generated by using the **Generate Code** button.

3.7. Converting Device Initialization Project To Processor Expert

The project created using the Device Initialization can be converted to Processor Expert. This is useful when the user finds out that he/she would like to use additional features of Processor Expert. Please see the chapter *1.4 Rapid Application Development Tools* for the tools comparison. This conversion is available for C projects only.

Warning: Don't forget to backup the whole project before the conversion. Some files will have to be removed from the project. The conversion to Processor Expert is recommended to experienced users only.

Conversion Steps

1. Generate code to save the last state of the design (if you haven't already done it).
2. Backup the whole project.
3. Select the menu command **Processor Expert > Open Processor Expert for <projectname>.mcp**
4. Switch to **Files** tab of the CodeWarrior project panel.
5. Remove the following files from the project using the *DEL* key or the *Remove* pop-up menu command:
   - Sources / main.c *(It will be replaced by the {projectname}.c)*
   - Include / derivative.h *(It will be replaced by the generated IO_Map.h)*
   - Include / <*CPUderivative*>.h *(It will be replaced by the generated IO_Map.h)*
   - Linker Files / Project.prm, in case of ColdFire it's Project.lcf *(It will be replaced by the generated {projectname}.prm/.lcf)*
   - Libs / <*CPUderivative*>.C *(It will be replaced by the generated IO_Map.h)*
6. Remove all object code using the menu command **Project > Remove object code...**.
7. Switch to **processor Expert** tab of the CodeWarrior project panel.
8. Generate the code with the command **Processor Expert > Generate Code <projectname>**
9. Copy the user code from the function in *main.c* into the newly generated main module <*projectname*>.c at the place marked with the text */* Write your code here */*.
10. Copy the user ISRs code from the in *MCUinit.c* (or a filename specified in options.) into a new user

module or to the main module.

11. Make the project using the **Project Make** command.

4. Tutorials

The following tutorials are available:

The Device Initialization tutorials are available within the CodeWarrior tutorials. Please follow these instructions to start the tutorial:

- Invoke the CodeWarrior startup dialog by starting the CodeWarrior or select the **File | Startup dialog** main menu command.
- Click on the **Run Getting Started Tutorial** button.
- Select a tutorial tutorial from the tutorials contents.

5. Help Revisions History

The current revision number: **2.9** (Generated: 14.7.2010 09:12:14)

18.1.2010 Revision 2.9

- Minor content updates

30.11.2009 Revision 2.8

- Minor content updates

7.10.2009 Revision 2.7

- Language corrections

16.9.2009 Revision 2.6

- Added additionally modified registers

4.6.2009 Revision 2.5

- Added information regarding RS08 interrupt support

19.5.2009 Revision 2.4

- updates related to MPC512x

8.4.2009 Revision 2.3

- updated copyright information

16.12.2008 Revision 2.2

- updated Converting to PE

9.9.2008 Revision 2.1

- New version for HCS12(X)

10.4.2008 Revision 2.0

- Added name checking to DI to PE conversion chapter
- Removed version number from window title

29.11.2007 Revision 1.9

- Update options names
- Separated helps for individual releases.

10.08.2007 Revision 1.8

- Added __initialize_hardware function description

26.07.2007 Revision 1.7

- Changed handling of ISRs contend during generation

07.03.2007 Revision 1.6

- Initial addition of ColdFire V1

12.09.2006 Revision 1.5

- Updated Tutorials, Quick start and Code Genaration and usage to match the project wizard in CodeWarrior for HC(S)08 V5.1.
- Removed animated tutorials - replaced by links to the same tutorials available from startup dialog.
- Added a new chapter 'Converting project to use the Device Initialization'.

27.04.2006 Revision 1.4

- Added information on Initialization Value meaning.
- Added information on Absolute/relative assembly options.

21.11.2005 Revision 1.3

- Tutorials updated.
- Added animated versions of tutorials.
- The Quick Start and Code Generation chapter updated.

7.11.2005 Revision 1.2

- Corrected error in the CPU derivative name in tutorials

1.11.2005 Revision 1.1

- Languace corrections
- Updated the Options, Tutorial and PE conversion chapters.

26.10.2005 Revision 1.0

- Initial release.

INDEX

Adding Device Initialization 23
Changing CPU 22
Code generation 19
Component 18, 4
Converting project 23
Converting to Processor Expert 24
CPU component 4
Creating ISRs 21
Design 4
Design steps 5
Features 4
Initialization code 19
Inspector 4, 12
Inspector items 14
Internal peripherals 4
Interrupt initialization 21
ISRS 21
Main Menu 8
Module 4
Options 16
Peripheral Initialization 18
Peripheral settings 4
Porperty types 14
Project creation 5
Properties 5
RAD tools 6
Saving settings 21
Settings backup 21
Target CPU 5
Tools comparison 6
Using generated code 19